Document Number	ENG-25670
Revision	2.0
Author	Ulrica de Fort-Menares
	Ram Golla
Contributors	Dalen Bosteder
	Ulrica de Fort-Menares
	Ram Golla
	Dean Hiller
	Fan Jiao
	Dennis Lau
	Leo Pereira
	Sri Gundavelli
	Johnathan Trostle
	Hugh Wong
Project Manager	Ulrica de Fort-Menares

IOS CNS/AD Client

System Functional Specification

Project Headline

This document specifies the Cisco Networking Services client component on IOS platforms. This is an infrastructure project that provides the infrastructure for IOS applications to query and access data that resides in a Directory Server via LDAP V3.

Reviewers

Department	Name and Title
Development Engineering	Anson Chen, Director IOS Infrastructure Dalen Bosteder, IOS Protocols Ray Bell, Director Directory Services, NSMBU John Strassner, Chief Architect NSMBU Dalen Bosteder, Manager IP ISU Dalia Geller, Manager Security ISU
Product Mktg	Kurt Dahm, NSMBU Roger Wood, NSMBU
Program Management	Dave Berry, Program Manager NSMBU Ginny Vincenzini, Program Manager NSMBU
Customer Advocacy	N/A

Modification History

Rev	Date	Originator	Comment
1.0		Ulrica de Fort-Menares	Draft
2.0		Ram Golla	Added writeup for Unicode & UTF8 support

1.0Purpose

1.1 Scope

Rapid Internet growth has created the need for more robust, scalable and secure directory services. Cisco, through a strategic development relationship with Microsoft, has chosen Microsoft's Active Directory as the foundation of the Cisco Networking Services (CNS) product. Cisco Networking Services is focused on creating rich network management and provisioning services. Cisco is committed to enabling applications to leverage advanced network services using the directory in response to the needs of business-critical applications.

This document contains the requirements for the IOS CNS client feature and its architectural design. This project is a standard IOS infrastructure project that provides the infrastructure for IOS applications to query, access and update data that resides in a Directory Server via Lightweight Directory Access Protocol (LDAP). This project is not visible as a feature. IOS applications need to be developed to take advantage of the Cisco Networking Services. Directory services can be used for everything from user authentication to policy-based management. Some of the IOS applications that will make use of this services are:

- Network Management (eg. Auto-configuration of the router)
- QOS Management
- DHCP
- DNS
- Policy Server
- Security (IPSEC, security authentication, certification administration)
- H323 applications (Call Dial plan)
- CORBA (Naming Services and Trader services)
- Mustang (Dialer information for large scale call out)
- Cable modems
- APPN (resource registration)
- DLSw (MAC Address caching)

As more applications leverage the common data within directory services, the directory will convincingly become the logical place to bring all kind of key information together to make the network more intelligent and scalable.

1.2 Definitions

This section defines words, acronyms, and actions which may not be readily understood.

AD	Microsoft's Active Directory
API	Application Program Interface
APPN	Advanced Peer-to-Peer Network
ARF	Automated Regression Factory
CDSI	Cisco Directory Services Interface
CNS	Cisco Networking Services
CORBA	Common Object Request Broker Architecture
DHCP	Dynamic Host Configuration Protocol
DLSw	Data Link Switching
DNS	Domain Name System
FCS	First Customer Ship
GSS-API	Generic Security Service Application Program Interface
IETF	Internet Engineering Task Force
IOS	Internetwork Operating System
IPSEC	IP Security Protocol
PAC	Privilege Attribute Certificate
KDC	Key Distribution Centre
QOS	Quality of Service
RR	Resource Record
SASL	Simple Authentication and Security Layer
SDK	Software Development Kit
SRV	Service
SSL	Secure Socket Layer
SSPI	Security Support Provider Interface
TGT	Ticket-granting Ticket

2.0Functional Overview

2.1 System Overview

This project implements LDAP V3 clients plus enhancements on IOS platforms. The feature is platform independent and it should function in all platforms.

LDAP support will enable the routers and switches to communicate with any vendors directory to discover information stored on the directory.

2.2 Feature List

A brief description of each feature for the IOS CNS Client is listed below:

2.2.1 LDAP V3

LDAP V3 supports all protocol elements of LDAP V2 (RFC 1777) and offers significant enhancements over LDAP V2. The following is a list of the enhancements:

- LDAP V3 supports a referral capability so one directory server can forward a client's query to another.
- LDAP V3 supports schema discovery, so an LDAP client can learn about the structure of the information in a directory. Because LDAP must be able to search, read and update server information on behalf of the client, the client must have prior knowledge of the directory's schema, or have some facility for discovering and interpreting schema.
- Normally a server returns all entries resulting from a search to the client. The client has no ability to regulate this flow. LDAP V3 supports paging where search results can be retrieved a page at a time.
- The protocol can be extended to support new operations, and controls may be used to extend existing operations.
- Attribute values and Distinguished Names have been internationalized through the use of the ISO 10646 character set.
- The weak security provided in LDAP V2 is one of its most serious shortcomings. This is addressed by LDAP V3. It supports stronger and more extensible authentication mechanisms. SASL mechanisms may be used with LDAP to provide association security services.

2.2.2 Locator Services

Microsoft's Active Directory has a concept of multi-master replication where there are 'mirror' directory servers for fault-tolerance, scalability and geographic distribution. A technique is required for a directory client to locate the closest directory server in the network. One method is to use a round-robin fashion to distribute the load among these geographically dispersed directory servers, but this technique is not always scalable and does not guarantee that end users will experience an acceptable level of service.

2.2.2.1 Microsoft's Locator implementation

Microsoft provides a mechanism to locate the closest directory server in the network. Instead of knowing the name or the IP address of the directory server, an LDAP client askes for a specific service/protocol as specified in RFC 2052. The LDAP client gets back the names of all the available servers with LDAP services. For a detailed description of Microsoft's Locator implementation, refer to [26] in the References section.

Figure 1 below depicts the scenario.

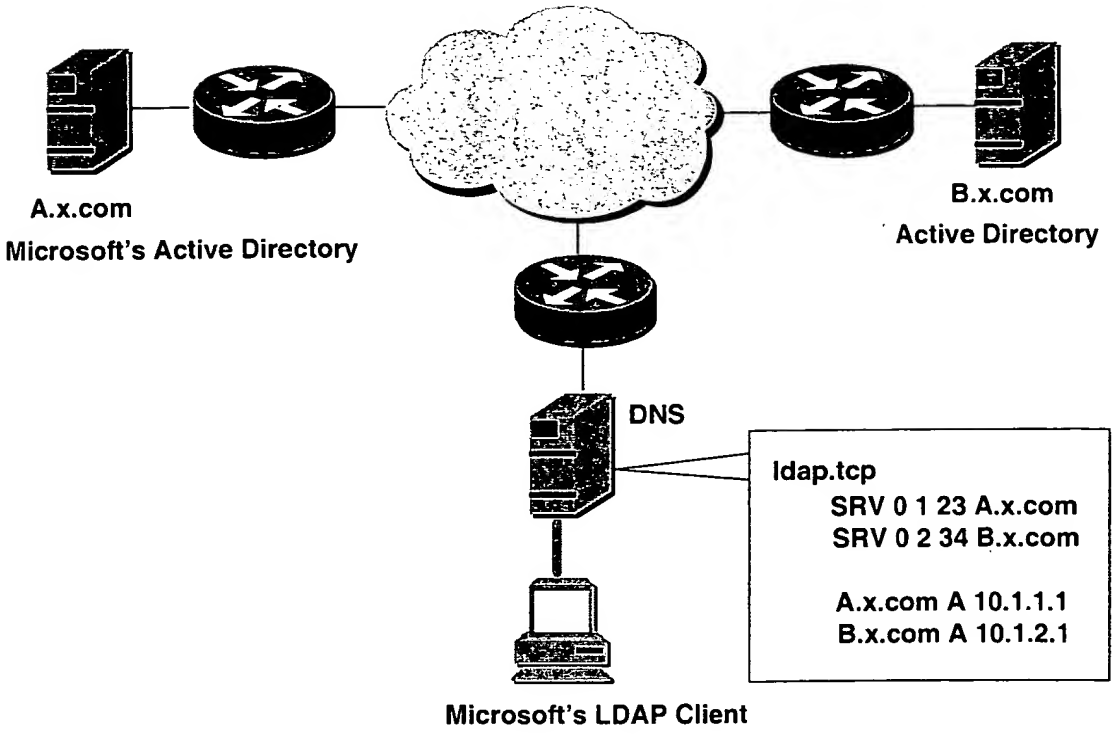

Figure 1 Microsoft's Locator Service

During the Active Directory install process, Active Directory will dynamically add the SRV and the A records to the primary DNS as shown in Figure 1 above.

The following shows the DNS SRV records:

ldap.tcp.<DnsDomainName>.

ldap.tcp.<SiteName>.sites.<DnsDomainName>.

ldap.tcp.pdc.ms-dcs.<DnsDomainName>.

ldap.tcp.fdc.ms-dcs.<DnsDomainName>.

ldap.tcp.gc.ms-dcs.<DnsDomainName>.

ldap.tcp.<SiteName>.sites.gc.ms-dcs.<DnsTreeName>.

ldap.tcp.<DomainGuid>.domains.ms-dcs.<DnsTreeName>.

ldap.tcp.writable.ms-dcs.<DnsDomainName>.

ldap.tcp.<SiteName>.sites.writable.ms-dcs.<DnsDomainName>.

The LDAP client in Figure 1 does a service lookup of ldap.tcp.x.com. (Note: only Microsoft's LDAP client supports the service lookup today.) The DNS returns the two SRV RRs which contain the names of the host that provide the LDAP service to the LDAP client. The LDAP client connects to one of the Directory Servers and obtains the site information about itself, the site information and the capability information about the Directory Server. Based on that information, the LDAP client can determine the closest server. The LDAP client performs a DNS lookup to obtain the site information and connects to the preferred Directory server.

This mechanism is administrative intensive since the site information has to be manually defined. The Cisco DistributedDirector provides a better solution.

2.2.2.2 Cisco DistributedDirector

existing product

Figure 2 below depicts the scenario with the Cisco DistributedDirector deployed.

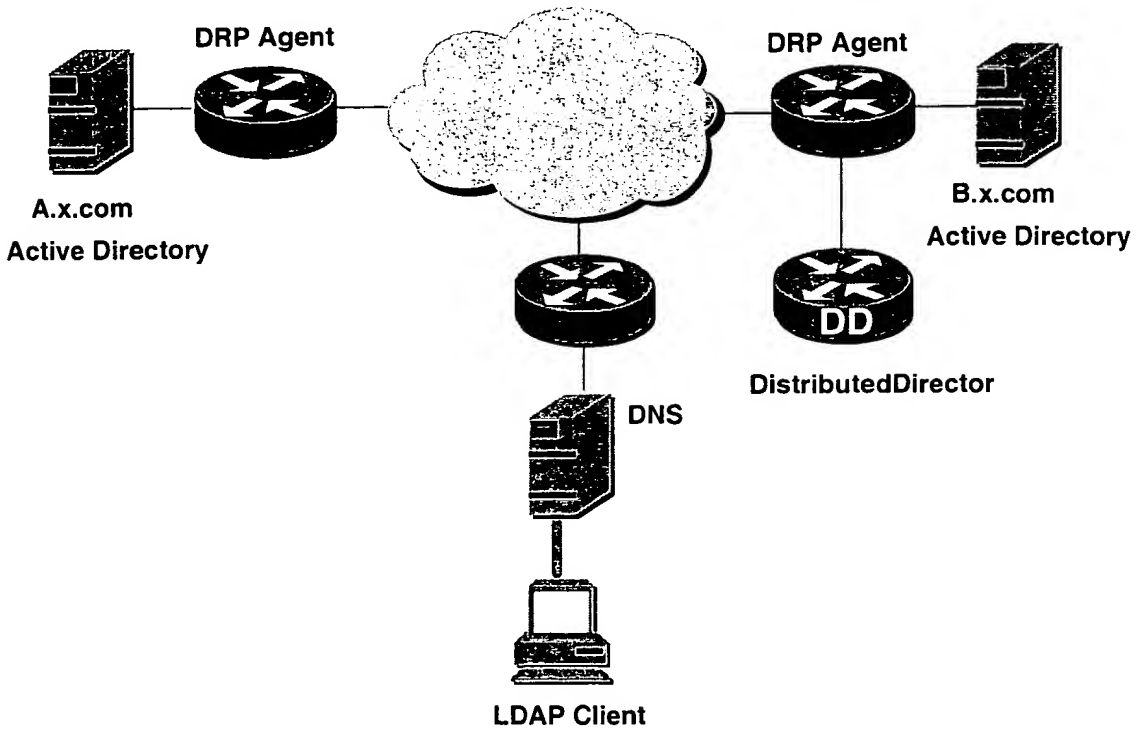

Figure 2 Cisco DistributedDirector

If the Cisco DistributedDirector is used, the administrator **must** configure the DD as the primary DNS for the Active Directory. During the Active Directory install process, Active Directory will attempt to dynamically add the SRV records to the DD. Since the DD does not support dynamic DNS update (RFC 2136), the SRV records will not be added to the DD. The SRV records have to be manually added to the DD.

The LDAP client in Figure 2 does a service lookup of ldap.tcp.x.com to its DNS. Since the SRV records cannot be found, the DNS refers to the DistributedDirector as the authoritative nameserver. The DNS queries the DistributedDirector by forwarding the service lookup of ldap.tcp.x.com. The DistributedDirector accepts the service lookup. It leverages the intelligence in the network by

determining the client-to-server topological proximity and it measures the round trip delay time to automatically, dynamically and efficiently pick the best server for the LDAP client. The DistributedDirector returns to the DNS a single IP address corresponding to the best server. The DNS forwards this result to the LDAP client.

The DistributedDirector is completely transparent to the LDAP client and the DNS. This means that the Microsoft Locator mechanism and the DistributedDirector can coexist. For a detailed description of the DistributedDirector, refer to the References section at the end of the specification.

Advantages:

- The DistributedDirector eliminates the need for LDAP clients to choose a server from a list of possible servers.
- The administrative overhead of defining site information associated with the Microsoft implementation is eliminated.
- The Cisco DistributedDirector uses routing table intelligence in the network infrastructure and it uses client-to-server link latency to determine the round-trip times. This method is far more sophisticated and dynamic than Microsoft's method.

Disadvantages:

- The Cisco DistributedDirector is packaged as a software/hardware bundle. It runs on a stand-alone C2500 or a C4700 device. There is an additional cost for the customer. Note: only one DistributedDirector is required per domain. The cost issue becomes less significant in a large network.
- There is minimal administrative overhead associated with the Cisco DistributedDirector.

Dependencies/Limitations

- IP addresses returned by the DistributedDirector are assigned a default TTL of zero seconds. The DNS must pay attention to the TTL record for a valid IP address to prevent local DNS caching.
- The DistributedDirector must support service lookup as specified in RFC 2052. The DistributedDirector will be enhanced in 12.0(2)T to support SRV RR.
- Currently, the DistributedDirector supports a maximum of 8 replicas. This limitation will be removed in the next release.

Recommendations:

It is recommended that the Locator API be called before the LDAP_INIT API. Consider the following scenarios:

1) Directory Server = Active Directory

If there is the Cisco DistributedDirector in the network, the IOS CNS client will issue a service lookup of LDAP to its DNS. The DistributedDirector will return a single IP address which is the closest directory server in the network.

If the Cisco DistributedDirector does not exist in the network, the Microsoft's Locator implementation will be used to return the closest directory server in the network.

2) Directory Server = non Active Directory Server

If the Locator API is used, the IOS CNS client will issue a service lookup of LDAP to its DNS. This request will fail. Next, the IOS CNS client will issue a DNS resolve for the IP address.

Note: The Locator API is optional, but recommended to take full benefit of the locator services provided by the IOS CNS/AD client.

2.2.3 Event Services

CNS/AD Event Services was intended to satisfy a set of business requirements. Refer to [5], [6] and [7] in the references section for more details.

2.2.4 Security Interface

Security is essential for controlling access to the directory servers. LDAP V3 provides simple authentication using a cleartext password as well as any Simple Authentication and Security Layer (SASL) mechanism. SASL is a layered architecture for using different security providers.

Although simple authentication is supported in LDAP V3, use of cleartext password is discouraged as it cannot guarantee confidentiality. For the IOS CNS client, Kerberos V5 is the default authentication security protocol. The IOS implementation of Kerberos is based on the code from CyberSafe.

Figure 3 below shows how an IOS application uses Kerberos V5 as the security protocol for authentication when communicating with the directory server.

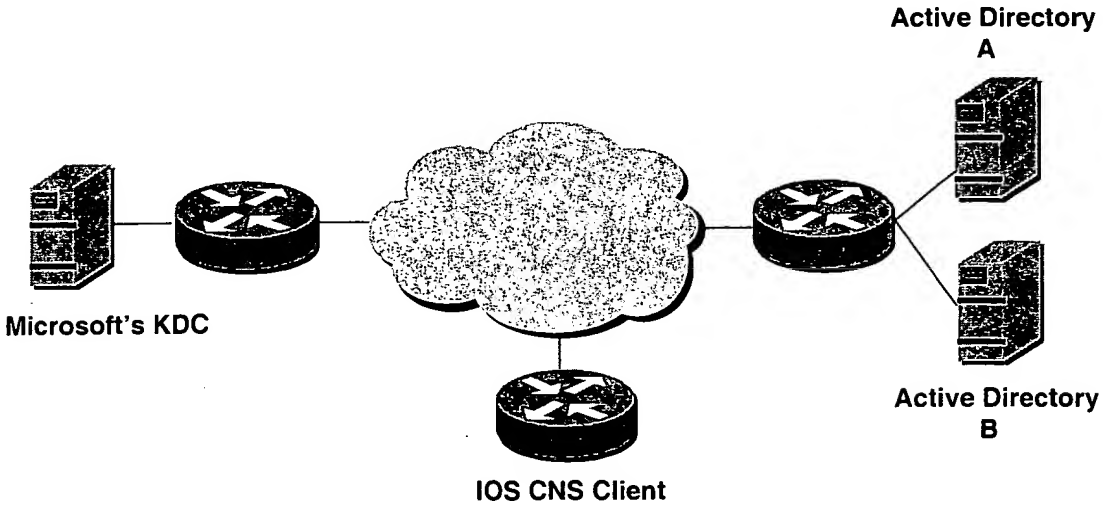

Figure 3 IOS CNS Kerberos Security

When the IOS CNS client starts up, it obtains a Kerberos Ticket-granting ticket (TGT) and a session key from its KDC. This information is stored in memory.

When an LDAP application needs to access Active Directory A, the IOS CNS client sends the TGT to the KDC along with a request for a ticket to A. The KDC sends IOS CNS a ticket to A. Now IOS CNS sends a request to Active Directory A consisting of the ticket to A. Active Directory A decrypts the ticket and discovers IOS CNS. The LDAP application is now ready to communicate with Active Directory A.

When a second LDAP application needs to access Active Directory A, the IOS CNS client can simply reuse the ticket to A. If the second LDAP application needs to access Active Directory B, the IOS CNS client needs to obtain from the KDC a ticket to B.

2.3 Features Not Addressed

The following features are not covered in the System Functional Specification because they are currently under design by NSMBU.

2.3.1 CNS Extension Libraries

This includes providing a set of libraries to parse octet strings. The design for the CNS Extension API is underway by NSMBU. The work on IOS CNS Extension API will be scheduled based on the dates provided by NSMBU.

2.3.2 Unicode

Unicode is not supported in IOS.

2.4 Testability Considerations

- Regression test the applications that use the LDAP V2 API. Where possible, use the existing LDAP V2 testbed for regression testing LDAP V3.
- Enhance the Cisco LDAP V2 test scripts to support LDAP V3.
- Integration testing with all of Cisco's CNS Server platforms to ensure interoperability.
- Performance and stress test the IOS CNS client code.
- Interoperability testing with an LDAP V2 and an LDAP V3 Directory Server.
- Automate the test scripts and submit to ARF.

2.5 Network Management Considerations

This section determines how the feature will be managed via the command line interface.

2.5.1 CLI

No CLI will be provided with LDAP since it is a library. LDAP applications will provide the CLI interface.

2.5.2 Debug

debug ldap <all / connect / bind / request / receive data / ber / referrals / errors >

all - turn on all the debugging

connect - debug the opening of the connection

bind - debug the ldap bind

request - debug the ldap request

receivedata - debug the data received

ber - debug the ldap ber encoding

referrals - debug the referrals

errors - debug all ldap errors

3.0Internal Specifications (Product Architectural Design)

3.1 Overview

Figure 4 gives a high level description of major software components for the IOS CNS client.

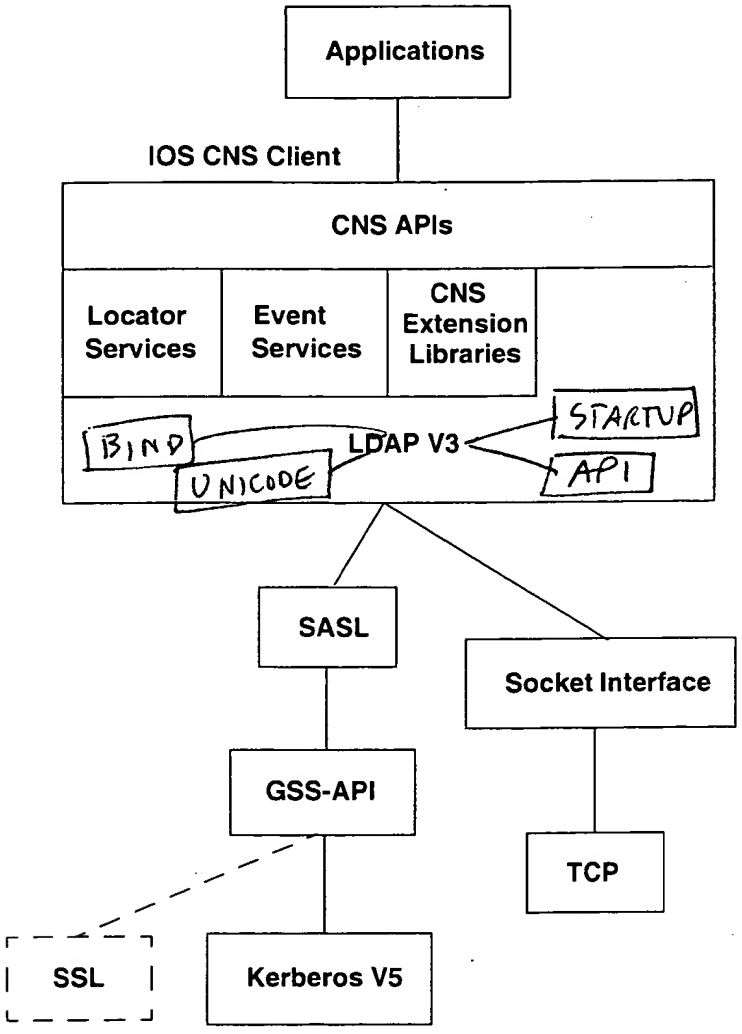

Figure 4 IOS CNS Client Components

The IOS CNS client project is a standard IOS infrastructure project that provides the infrastructure for IOS applications to query and access data that resides in a Directory Server via Lightweight Directory Access Protocol. This project is not visible as a feature. IOS applications must be developed to take advantage of the Cisco Networking Services.

LDAP is an Internet standard defined by the IETF that provides a common means to access directory services by diverse clients. The emergence of standards is providing much-needed vendor interoperability, freeing developers from having to tie their applications or services to a particular directory. By implementing LDAP, a Cisco device can interoperate with any directory servers in the network.The standard is described in RFC 2251. LDAP uses TCP as the underlying transport and it uses the IOS socket interface.

Security is essential for controlling access to the directory servers. LDAP V3 provides simple authentication using a cleartext password as well as any Simple Authentication and Security Layer (SASL) mechanism. SASL is a layered architecture for using different security providers. SSPI is the principal provider. Use of cleartext password is discouraged as it cannot guarantee confidentiality. In the first release, Kerberos V5 is the default security protocol for LDAP using GSS-API as the security interface. Security technologies are changing rapidly. As other security protocols evolve, such as public-key certificates, SSL may be considered for future releases.

The LDAP Client V3 code is based on Microsoft's source code. Porting of this code is a key part of this project. In addition, there are a number of Cisco enhancements. They are Locator Services, CNS extension APIs and Event Services.

Locator Services provides a scalable solution for an LDAP client to locate the closest Directory Server in the network. Microsoft provides a rather inefficient mechanism that is administrative intensive for locating the closest Directory Server in the network. Cisco uses the sophisticated DistributedDirector which leverages the intelligence in the network to automatically, dynamically, and efficiently pick the best server for the client.

CNS Extension libraries are a set of libraries that map Cisco specific schema information to octet strings, and vice versa.

Event Services is a CNS component that provides a mechanism for a directory client to register to its server events that are of interest to him. When the event occurs, the directory server notifies the clients so that the client can take appropriate actions.

3.2 Major Components

3.2.1 LDAP V3

The integration of the Microsoft's LDAP V3 code to the IOS environment will be the focus of this component. The intent is to leave the Microsoft's source code intact.

3.2.1.1 Startup

LDAP is a separate subsystem and it consists of a set of libraries. The subsystem initialises the global variables.

Assumption:

Before an ldap application starts up, the router should have obtained the Kerberos TGT and the session key from the KDC.

3.2.1.2 API

The full set of LDAP APIs will be supported on IOS. For low end platforms with memory constraints, a subset of the LDAP APIs will be implemented. The subsets are broken down into:

- Synchronous vs. Asynchronous (Note: The names of the synchronous functions always end in _s.)
- Extension (Note: The names of the Extension functions always end in _ext) vs. Non-extension operations

The table below is a summary of the LDAP V3 APIs. For a detailed description, refer to [21] in the Reference section.

Table 1: LDAP V3 API

API Name	Description
ldap_abandon, ldap_abandon_ext	Abandon an LDAP operation in progress
ldap_add, ldap_add_s, ldap_add_ext, ldap_add_ext_s	Add entries to the LDAP directory
ldap_close_extended_op	Close extended requests / operations
ldap_compare, ldap_compare_s, ldap_compare_ext, ldap_compare_ext_s	Compare string attribute value and binary controls
ldap_connect	Takes in the LDAP handle returned from ldap_init() and connects to the server
ldap_control_free, ldap_controls_free	Dispose control(s)
ldap_count_entries	Count the number of entries returned
ldap_count_references	Count the number of references returned
ldap_count_values, ldap_count_values_len	Count the number of values returned by ldap_get_values() / ldap_get_values_len()
ldap_create_page_control	Construct page control
ldap_create_sort_control	Construct sort control
ldap_delete, ldap_delete_s, ldap_delete_ext, ldap_delete_ext_s	Delete entries from the LDAP directory
ldap_dn2ufn, ldap_ufn2dn	Converts DN into user friendly format and reverse
ldap_escape_filter_elem	Takes any filter element and adds necessary escape characters
ldap_explode_dn, ldap_explode_rdn	Break up the name into its component parts
ldap_extended_operation, ldap_extended_operation_s	Extend LDAP operations
ldap_first_attribute, ldap_next_attribute	Step through the list of attribute types returned with an entry
ldap_first_entry, ldap_next_entry	Step through a set of entries in a search result
ldap_first_reference, ldap_next_reference	Step through and retrieve the list of continuation references in a search result
ldap_get_dn	Retrieve the name of an entry

Table 1: LDAP V3 API

API Name	Description
ldap_get_option, ldap_set_option	Access or update the current value of various session-wide parameters
ldap_get_values, ldap_get_values_len	Retrieve the values of a given attribute from an entry
ldap_get_next_page, ldap_get_next_page_s	Retrieve next result page from the whole result set
ldap_get_paged_count	Count number of page in the whole result set
ldap_init, ldap_sslinit	Initialize and LDAP session
ldap_memfree	Dispose of memory allocated by an LDAP API
ldap_modify, ldap_modify_s, ldap_modify_ext, ldap_modify_ext_s	Modify an existing LDAP entry
ldap_msgfree	Frees the results obtained from a previous call to ldap_result()
ldap_msgtype	Retrieve the type of LDAP message
ldap_msgid	Retrieve the message ID from a LDAP message
ldap_parse_page_control	Parsing the control
ldap_parse_reference	Return the list of subordinate referrals in a search response message
ldap_parse_result, ldap_parse_extended_result	Extract information from results by other LDAP APIs
ldap_parse_sort_control	Parsing the sort control returned by the server
ldap_rename_ext, ldap_rename_ext_s	Modify DN protocol operation
ldap_result	Obtain the result of a previous asynchronously initiated operation
ldap_sasl_bind, ldap_sasl_bind_s, ldap_simple_bind, ldap_simple_bind_s	Used to authenticate to the directory
ldap_search, ldap_search_s, ldap_search_ext, ldap_search_ext_s	Search the LDAP directory
ldap_search_init_page	Set up a request buffer to hold the information for the search
ldap_search_abandon_page	Abandon a sequence of paged search requests
ldap_unbind, ldap_unbind_s	Terminate LDAP session
ldap_value_free, ldap_value_free_len	Retrieve the values of a given attribute from an entry
LdapGetLastError	Thread Safe way to get last error code returned by LDAP API

Table 1: LDAP V3 API

API Name	Description
LdapUTF8toUnicode, LdapUnicodeToUTF8	Convert UTF strings to Unicode and Unicode string to UTF strings
ldap_check_filter	Check the validity of search filter. One of the ldap_search helper functions
ber_alloc_t	Constructs and returns BerElement
ber_bvdup	Return a copy of a berval
ber_bvecfree	Frees an array of bervals returned
ber_bvfree	Frees a berval returned
ber_first_element	Traverse a SET, SET OF, SEQUEMCE, or SEQUENCE OF data value
ber_flatten	Allocate a struct berval
ber_free	Frees a BerElement
ber_init	Constructs a BerElement and returns a new BerElement containing a copy of the data in the bv argument
ber_next_element	Position the state at the start of the next element in the constructed type
ber_peek_tag	Return the tag of the next element to be parsed in the BerElement argument
ber_printf	Encode a Ber element
ber_scanf	Decode a BER element
ber_skip_tag	Similar to ber_peek_tag, except that the state pointer in the BerElement argument is advanced past the first tag and length, and is pointed to the value part of the next element

3.2.1.3 Bind Operation

The function of the Bind operation is to initiate a protocol session between a client and a server, and to allow the authentication of the client to the server. The Bind operation must be the first operation request received by a server from a client in a protocol session.

An application connects to the server by issuing an ldap_bind request, the IOS CNS client opens a socket and a TCP connection is established between the client and the server. An application can then issues other operations using the same socket. When another ldap application needs to issue an ldap-request, even to the same server, the application will again issue an ldap_bind request which also triggers the IOS CNS client to open another socket and a separate TCP connection is established for that new application. The number of TCP connections is equal to the number of ldap_bind requests.

3.2.1.4 Unicode and UTF8 Support

Introduction to Unicode and UTF8

Unicode is a 16 bit character encoding standard, aligned with ISO 10646, that represents most of the characters used in general text interchange throughout the world. 16-bit characters, however, are not compatible with many current applications and protocols, and this has led to the development of a few so-called UCS transformation formats (UTF), each with different characteristics. UTF-8 has the characteristic of preserving the full US-ASCII range.

UTF8 encoding uses a string of bytes to represent a 16-bit Unicode string where as ASCII text (<=U+007F) remains unchanged as a single byte, U+0080-07FF including Latin, Greek, Cyrillic, Hebrew, and Arabic) is converted to a 2-byte sequence, and U+0800-FFFF (Chinese, Japanese, Korean, and others) becomes a 3-byte sequence. The advantage is that most ASCII text remains unchanged and almost all editors can read it.

Character Encoding for IOS-LDAP API

One of the key enhancements in LDAP v3 is the support for international character sets by means of utf8 encoding. Object names and certain string attributes in Active Directory can contain non ASCII characters. This is probably not an issue for most of the IOS applications, since they may not be manipulating the content. However some IOS applications and services may need to manipulate these objects and attributes. If the application passes these strings to standard string libraries will not function properly. Following utf8 functions need to be provided so that applications expecting international character strings can handle them properly.

Standard string functions such as strcpy, strcat, strcmp will function properly with utf8 strings. Following minimal function set need to be provided so that IOS applications can handle nternational characters properly.

//utf8 characters can be 1, 2 or 3 bytes long

Utf8CharLen - Returns the length of the first character (possible values 1, 2 or 3)

//regular stricmp & strlen will fail for non ascii characters

Utf8Stricmp - Compares the two strings, ignoring the case.

Utf8StringLen - Returns the number of characters in a string.

//avoid regular character pointer increment

Utf8NextChar - Returns the pointer to the next character immediately after the given string position
Utf8PrevChar - Returns the pointer to the previous character immediately before the given string position.

3.2.2 Locator Services

The Locator Services client will use the IOS DistributedDirector to locate the closest Directory server in the network. If the DistributedDirector is not used in the customer's network, the Microsoft's Locator implementation will be supported.

Locator API - DsGetDcName API

The DsGetDcName API returns the name of a Domain Controller (DC) in a specified domain. The domain may be trusted (directly or indirectly) or untrusted by the caller. DC selection criteria is supplied to the API to indicate preference for a DC with particular characteristics.

The DsGetDcName API returns the name of a Domain Controller (DC) in a specified domain. The domain may be trusted (directly or indirectly) by the caller or may be untrusted. DC selection criteria are supplied to the API to indicate preference for a DC with particular characteristics.

The DsGetDcName API does not require any particular access to the specified domain. By default, DsGetDcName does not ensure the returned domain controller is currently available. Rather, the caller should attempt to use the returned domain controller. If the domain controller is indeed not available, the caller should repeat the DsGetDcName call specifying the DS_FORCE_REDISCOVERY flag.

DWORD

DsGetDcName(

LPCTSTR ComputerName **OPTIONAL,**

LPCTSTR DomainName **OPTIONAL,**

GUID *DomainGuid **OPTIONAL,**

LPCTSTR SiteName **OPTIONAL,**

ULONG Flags,

PDOMAIN_CONTROLLER_INFO *DomainControllerInfo

)

Parameters:

ComputerName - **Not supported on IOS/UNIX.** (Specifies the name of the server to remote this API to. Typically, this parameter should be specified as NULL.)

DomainName - The name of the domain to query. This name can either be a DNS-style name (e.g., cisco.com.) or a flat-style name (e.g., cisco). If a DNS-style name is specified, the name may be specified with or without a trailing period is specified.

If NULL is specified and the DS_GC_SERVER_REQUIRED flag is specified, the tree name of the primary domain of *localhost* is used.

If NULL is specified and the DS_GC_SERVER_REQUIRED flag is not specified, the domain name of the primary domain of *localhost* is used.

DomainGuid - Specifies the Domain GUID of the domain being queried. This value is used to handle the case of domain renames. If this value is specified and *DomainName* has been renamed, DsGetDcName will attempt to locate a DC in the domain having this specified *DomainGuid.*

SiteName - Specifies the name of the site the returned DC should be "close" to. The parameter should typically be the name of the site the client is in. If not specified, the *SiteName* defaults to the site of *ComputerName*.

Flags - Passes additional information to be used to process the request. *Flags* can be a combination of the following values bitwise or'ed together:

Flags Values

DS_FORCE_REDISCOVERY - Requires that the "closest" Domain Controller be determined again even though one is currently known in cache. This flag can be used in the case that an additional Domain Controller comes available or where a Domain Controller has been detected to be unavailable. The returned Domain Controller is verified to be running if this flag is specified. Without this flag, this API will only guarantee that the returned DC when the DC was initially entered into the cache.

DS_DIRECTORY_SERVICE_REQUIRED - **Not Supported on Unix/IOS.** (Requires that the returned DC support Directory Server API (i.e., runs NT 5.0 or later).)

DS_DIRECTORY_SERVICE_PREFERRED - **Not Supported on Unix/IOS.** (Prefers that the returned DC support Directory Service API (i.e., runs NT 5.0 or later). If no such DC is available, a prior version DC will returned. If no DC supporting a DS is available, DsGetDcName will return the name of the "closest" non-DS DC; however, DsGetDcName will only return the non-DS DC information after the attempt to find a DS DC has timed out.)

DS_GC_SERVER_REQUIRED - Requires that the returned DC be a Global Catalog (GC) server for the tree of domains with this domain as the root.

This flag may not be set if any of the following flags are set: DS_WRITABLE_REQUIRED, DS_FDC_REQUIRED or DS_PDC_REQUIRED.

DS_PDC_REQUIRED - Requires that the returned DC be the Primary Domain Controller for the domain.

This flag may not be set if any of the following flags are set: DS_WRITABLE_REQUIRED, DS_FDC_REQUIRED or DS_GC_SERVER_REQUIRED.

DS_WRITABLE_REQUIRED - Requires that the returned DC host a writable copy of the DS (or SAM). If the specified *DomainName* is a flat name, this flag is the same as DS_PDC_REQUIRED. If the specified *DomainName* is a DNS name, this flag finds DCs that advertise themselves as writable. An NT 5.0 "mixed mode" domain only advertises the PDC as writable. An NT 5.0 "non-mixed mode" domain advertises all of the DCs as writable. In future releases of NT, certain DCs may be configured as "read only", such DCs will not advertise themselves as writable.

This flag may not be set if any of the following flags are set: DS_PDC_REQUIRED, DS_FDC_REQUIRED or DS_GC_SERVER_REQUIRED.

DS_FDC_REQUIRED - **Not supported on IOS/Unix.** (Requires that the returned DC be the Floating Domain Controller for the domain. In a mixed domain with an NT 3.x or NT 4.x PDC and one or more NT 5.x BDCs, one of the NT 5.x BDCs plays the special role of replicating account changes from the PDC. That DC is called the "Floating" DC since the function "floats" to another NT 5.x BDC automatically if the current FDC becomes unavailable.
This flag may not be set if any of the following flags are set: DS_PDC_REQUIRED, DS_WRITABLE_REQUIRED or DS_GC_SERVER_REQUIRED.)

DS_IP_REQUIRED – Requires that the IP address of the discovered DC be returned in the *DomainControllerAddress* field.

DS_KDC_REQUIRED - **Not supported on IOS/Unix.** (Requires that the returned DC be currently running the Kerberos Key Distribution Center Service.)

DS_TIMESERV_REQUIRED – **Not supported on IOS/Unix.** (Requires that the returned DC be currently running the Windows Time Service.)

DS_IS_FLAT_NAME - **Not Supported on Unix/IOS.** (Specifies that the *DomainName* parameter is a flat name. As such, the (Section) will not be attempted. This flag may not be specified with the DS_IS_DNS_NAME flag. It is valid to set neither DS_IS_FLAT_NAME nor DS_IS_DNS_NAME; however, DsGetDcName() will take longer to find a DC in that case since it has to try both the DNS-

style and flat name. It is potentially ambiguous to specify neither flag. For instance, if you specify a domain name of "redmond", a domain with a flat name of "redmond" exists on you network, and a *different* domain with a DNS-style name of "redmond" exists on your network, then DsGetDcName() may potentially find a DC in either domain.)

DS_IS_DNS_NAME - **Always true on Unix/IOS.** (Specifies that the *DomainName* parameter is a DNS name. This flag may not be specified with the DS_IS_FLAT_NAME flag.)

DomainControllerInfo - Returns a pointer to a DOMAIN_CONTROLLER_INFO structure describing the domain controller selected.

Error Codes:

NO_ERROR - Operation completed successfully;

ERROR_NO_SUCH_DOMAIN: No DC is available for the specified domain or the domain does not exist.

ERROR_INVALID_DOMAINNAME - The format of the specified *DomainName* is invalid.

ERROR_INVALID_COMPUTERNAME - The format of the specified *ComputerName* is invalid.

ERROR_INVALID_FLAGS - The *Flags* parameter has conflicting or superfluous bits set.

ERROR_NOT_ENOUGH_MEMORY - There was not enough memory to complete the operation.

Various Winsock errors.

The DOMAIN_CONTROLLER_INFO structure is defined as follows:

typedef struct **_DOMAIN_CONTROLLER_INFO** {

LPTSTR *DomainControllerName;*

LPTSTR *DomainControllerAddress;*

ULONG *DomainControllerAddressType;*

GUID *DomainGuid;*

LPTSTR *DomainName;*

LPTSTR *TreeName*;
ULONG *Flags*;

LPTSTR *DcSiteName;*

LPTSTR *ClientSiteName;*

} **DOMAIN_CONTROLLER_INFO, *PDOMAIN_CONTROLLER_INFO**;

DOMAIN_CONTROLLER_INFO Field Definitions:

DomainControllerName - Pointer to a zero terminated string specifying the computer name of the discovered domain controller. The returned computer name is prefixed with \\. The DNS-style name (e.g., \\phoenix.microsoft.com.) will be returned if available. If a DNS-style name is returned, it will be terminated be a period indicating that the returned name is an absolute (non-relative) DNS name.

DomainControllerAddress - Pointer to a zero terminated string specifying the address of the discovered domain controller. The address is prefixed with \\ to aid in passing it as the ComputerName parameter to various subsequent API. This string will be one of the types defined by *DomainControllerAddressType.*

DomainControllerAddressType - Indicates the type of address specified in *DomainControllerAddress*. The list below is presented in order in case multiple forms are known. Valid values are:

DS_INET_ADDRESS - Address is a string-ized IP address (e.g., \\157.55.94.74) of the domain controller.

DomainGuid - The Domain GUID of the domain. This field will be zero if the DC does not have Domain GUID (e.g., the DC is not an NT 5 DC).

DomainName - Pointer to a zero terminated Unicode string specifying the domain name of the domain. The DNS-style name (e.g., microsoft.com.) will be returned if available. Otherwise, the flat-style name (e.g., microsoft) will be returned. This name may be different than the requested domain name if the domain has been renamed. If a DNS-style name is returned, it will be terminated be a period indicating that the returned name is an absolute (non-relative) DNS name.

TreeName - Pointer to a zero terminated Unicode string specifying the domain name of the domain at the root of the DS tree. The DNS-style name (e.g., microsoft.com.) will be returned if available. Otherwise, the flat-style name (e.g., microsoft) will be returned. If a DNS-style name is returned, it will be terminated be a period indicating that the returned name is an absolute (non-relative) DNS name.

Flags - Flags describing the domain controller.

DS_PDC_FLAG - Domain Controller is the PDC of the domain.

DS_WRITABLE_FLAG – Domain Controller host a writable DS (or SAM).

DS_FDC_FLAG - Domain Controller is the FDC of the domain.

DS_GC_FLAG - Domain Controller is a GC server for *TreeName*.

DS_DS_FLAG - Domain Controller is a Directory Service server for the domain.

DS_KDC_FLAG – Domain Controller is a Kerberos Key Distribution Center for the domain.

DS_TIMESERV_FLAG – Domain Controller is running the Windows Time Service for the domain.

DS_DNS_CONTROLLER_FLAG - *DomainControllerName* is DNS format.

DS_DNS_DOMAIN_FLAG - *DomainName* is in DNS format.

DS_DNS_TREE_FLAG - *TreeName* is in DNS format.

DcSiteName – The name of the site the Domain Controller is in. This value may be NULL if the DC is not in a site (e.g., the DC is an NT 4.0 DC).

ClientSiteName – The name of the site *ComputerName* is in. This value may be NULL if site the computer named by *ComputerName* cannot be determined (e.g., if the DS administrator has not associated the subnet the computer is in with a site).

3.2.3 Event Services

Please refer to [17] for more details.

---

1. This name should be accepted by all API calls that accept a ServerName parameter. However, certain API calls that existed prior to NT 5 should be passed DomainControllerName instead. For instance, NT 5 DCs support multiple domains. The DC is assigned multiple DNS host names, but a single IP address. An API, such as NetWkstaGetInfo or NetUser*, behaves differently depending on which domain is being used. If these API are passed DomainControllerAddress and it is an DS_INET_ADDRESS, they will respond for the default domain for the DC and not necessarily the desired domain.

3.2.4 Security Interface

The Microsoft LDAP V3 source code uses GSS-API as the interface for network security services. This includes porting of the GSS-API to provide security services.

Please refer to [30] for more details.

3.3 Major Internal Interfaces

Dependencies on other products:

3.3.1 Security Interface

LDAP V3 has to interface with Kerberos V5 via GSS-API. The Kerberos V5 feature has to be configured. Minimum Kerberos configuration:

kerberos local-realm <realm name>

kerberos realm <realm name>

kerberos server <Kerberos realm> <hostname | ip address> <port number>

kerberos credentials forward

3.3.2 Socket Interface

This involves mapping of the Microsoft socket API to the Cisco IOS Socket API.

3.4 Memory Size Implications

For security, the Kerberos code will add approximately 100k to the image size and the GSS-API code will add approximately ?k to the image size.

The LDAP V3 and Locator services code will add approximately 100k to the image size. The event services code will add approximately ?k to the image size.

4.0Issues, Risks, and Dependencies

4.1 Related Projects

CNS depends on a number of Kerberos enhancements implemented by the IOS Security Engineering group. They are the PAC format enhancements, TCP as an optional transport, remote downloading of the SRVTAB file, and the implementation of the GSS-API layer for CNS.

The use of the DistributedDirector to locate the best server also depends on an enhancement implemented by the IOS Protocols Engineering group. The DistributedDirector has to support the location of the server for a specific protocol and domain as described in RFC 2052. The DNS RR type is SRV RR.

4.2 Resource Contentions

An additional resource is required for this project. No resources have been assigned for testing the event services component.

4.3 Single Source Components

External Dependencies:

4.4 Technology Requirements

Stability of the Microsoft code is unknown.

The current IOS LDAP V2 code is approximately 20K. The initial code size for LDAP V3 is approximately 100K. Every effort will be made to reduce the code size, but it is not known at the time of writing how much it can be reduced. This may be a major issue for low end platforms.

References

[1] 'Cisco Directory Services Interfaces (CDSI) Functional Overview', John Strassner & Dave Berry, DSENG-021.1

[2] 'CDS Client for IOS System Functional Specification', Ram Golla, DSENG-009.7

[3] 'Cisco QoS Configurations & CDS Qos Schema System Functional Specification', Fan Jiao, DSENG-015.101

[4] 'Cisco Directory Services (CDS) Schema Specification', John Strassner, DSENG-010.4

[5] 'Cisco Directory Services Policy and Event Services Dictionary', John Strassner, J. J Ekstrom, Thomas McNeill, DSENG-011.001

[6] 'Cisco Directory Services Policy and Event Services Engine Requirements Specification', John Strassner, DSENG-012.001

[7] 'CDS Policy and Event Services Engine System Functional Specification', Thomas McNeill, DSENG-019.4

[8] 'Distributed Director: Software Unit Functional Specification', Paul Traina, Richard Johnson and Dhaval Shah, ENG-7562

[9] 'Cisco DistributedDirector', Keven Delgadillo, http://www.cisco.com/warp/public/751/distdir/dd_wp.htm

[10] 'Lightweight Directory Access Protocol API', Wen-Lin Tsao, ENG-15901

[11] 'IOS CNS/AD Client Program Plan', Ulrica de Fort-Menares, ENG-23055

[12] 'LDAP V3 Protocol Specification', RFC 2251

[13] 'LDAP V3 Attribute Syntax Definitions', RFC 2252

[14] 'LDAP V3 UTF-8 String Representation of Distinguished Names', RFC 2253

[15] 'LDAP V3 String Representation of LDAP Search Filters', RFC 2254

[16] 'LDAP V3 URL Format', RFC 2255

[17] 'LDAP V3 User Schema Summary', RFC 2256

[18] 'Lightweight Directory Access Protocol', RFC 1777

[19] 'The LDAP Application Program Interface', RFC 1823

[20] 'A DNS RR for specifying the location of services (DNS SRV)', RFC 2052

[21] 'The C LDAP Application Program Interface', Draft-RFC (draft-ietf-ldapext-ldap-c-api-00.txt)

[22] 'Dynamic Updates in the Domain Name System (DNS UPDATE)', RFC 2136

[23] 'Simple Authentication and Security Layer (SASL)', RFC 2222

[24] 'Generic Security Service Application Program Interface', RFC 1508

[25] 'The Kerberos Network Authentication Service (V5)', RFC 1510

[26] 'Locator Services Functional Specification', Sri Gundavelli, ENG-28610

[27] 'IOS CNS Client Event Services System Functional Specification', Fan jiao & Dennis Lau, ENG-28376

[28] 'IOS CNS/AD Client Software Unit Design Specification', Leo Pereira, ENG-25716

[29] 'IOS CNS/AD Client Test Plan', Hugh Wong, ENG-25717

[30] 'Software Unit Design Specification for GSS-API to support LDAP V3', Allen Long, ENG-29005

[31] 'IOS CNS Client Event Services Test Plan', Tony Zhang, ENG-28950

Attachments

As appropriate, examples of forms, log sheets, diagrams, schematics, or other pieces of information used in or generated when carrying out the activities of the document would be identified here and then attached as appendices to the procedure. (If attachments are added, delete this paragraph.)